Master's Thesis

Enhanced Reservoir Computing with Concatenation

Eonyoung Park

Department of Mathematical Sciences

Graduate School of UNIST

2020

Enhanced Reservoir Computing with Concatenation

Eonyoung Park

Department of Mathematical Sciences

Graduate School of UNIST

Enhanced Reservoir Computing with Concatenation

A thesis
submitted to the Graduate School of UNIST
in partial fulfillment of the
requirements for the degree of
Master of Science

Eonyoung Park

06/24/2020
Approved by

Advisor
Bongsoo Jang

Enhanced Reservoir Computing with Concatenation

Eonyoung Park

This certifies that the thesis of Eonyoung Park is approved.

06/24/2020

signature

Advisor: Prof. Bongsoo Jang

signature

Prof. Pilwon Kim : Thesis Committee Member #1

signature

Prof. Yunho Kim : Thesis Committee Member #2

Abstract

Recently, Reservoir Computing(RC) is spotlighted because of structural simplicity. RC is an architecture based on a Recurrent Neural Network(RNN) and the recurrent connections in RNN is called "reservoir" in RC. In this paper, we introduce the echo state network(ESN), which is one of the representative models of RC, and advanced versions of ESN.

We introduce the dynamic based RC with the Kuramoto model(RCK), which is a mathematical model used to describe synchronization. There are many modified versions of Kuramoto model, we use the explosive Kuramoto model in this paper. We test RCK using the tasks in the advanced version of ESN articles, and compare RCK to a simplified ESN.

We propose the concatenated RC. We apply the concatenation to RC, and compare the effect of concatenation using the capacity.

Contents

I Introduction

Recently, Reservoir Computing(RC), a Recurrent Neural Network(RNN)-based architecture, is spotlighted because of structural simplicity. RNN is a class of artificial neural network which has been used for temporal pattern recognition. Since RNN is pressured to be complex for the performance, the costs are increased exponentially. By this problem, RC receives attention. RC has the spatiotemporal complex dynamical structure, *reservoir*, so it is useful when we use the temporal(sequential) data, such as system data, handwritten digits. The advantage of RC compared to RNN is RC doesn't adjust the parameters in the process of making a reservoir. We obtain the dynamical patterns of reservoir by perturbing input data to reservoir. We analyze observed dynamical patterns with the parameter in the readout layer and deduct the output which we want. The main characteristic of reservoir computing is the parameter that is adjusted in the training step. Other neural networks adjust whole parameters in the network, but RC modifies the parameters only in a readout layer.

In this work, we introduce the RC. There are various types of reservoir computing such as Echo State Network [1, 2], Liquid-State Machine [3], Nonlinear Transient Computation [4]. Among them, the universally used method is Echo State Network(ESN), which is a method to analyze nonlinear behavior according to input data with a sparse connection. The advantage of the ESN is the stability of a model by the condition which is called echo state property(ESP). By this property, we don't need to consider the random variable. We introduce the modified versions of ESN for a prediction and reconstruction.

In the third section, we introduce the dynamics based RC using Kuramoto model. Kuramoto model is a mathematical model used to describe synchronization. It is a model for the behavior of a set of coupled chaotic oscillator. This model is proposed by Choi [5] in 2019, they use the Mackey-Glass equation for the test. We test the chaotic systems, such as Lorenz system and Rössler system, with the same parameters in several ESN articles and compare RCK to modified version in [2].

For the test, we use the tasks in the articles [2,6]. First one is reconstruction in [2]. Reconstruction is a process to recover the lost data with the correlated data. The condition of this process is a correlation between input and output data. For this task, we use the above chaotic systems and spatiotemporal chaotic equation(Kuramoto-Sivashinsky equation). Kuramoto-Sivashinsky equation [7, 8] is a fourth-order nonlinear partial differential equation, which derive the diffusive instabilities in the laminar flame front in the late 1970s. Second one is a real-time prediction in [6]. We bring the process of prediction in [6]. The process in the training period is the same as reconstruction, but the output data after the training period is used for the new input data of the next step.

In the last section, we introduce the concatenated reservoir computing. For the RCK, there is a limitation in the performance according to the size of reservoirs. To measure the performance, we compare the capacity which is a measurement of performance by using finite products of

polynomials. Capacity measures that how many delayed finite products of polynomials can be endured.

To improve the performance, we propose the reservoir computing with concatenation. We construct reservoirs with small nodes individually and concatenate these small reservoirs as one big reservoir. We expect lower costs and higher performance by the concatenation. We apply this concatenation to two reservoir computing and compare by the reconstruction.

II Preliminary

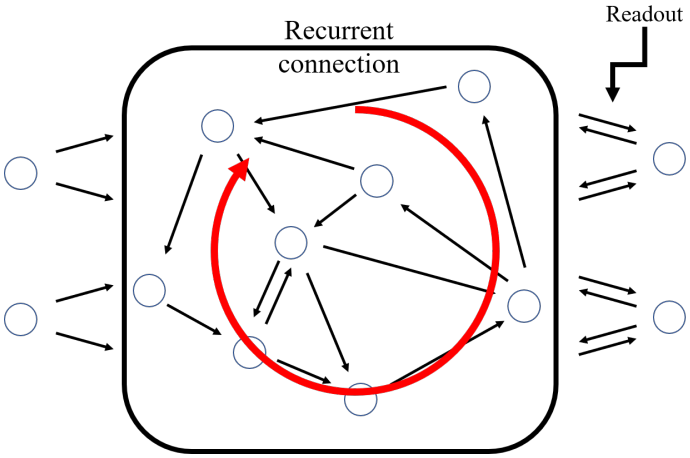


Figure 1: The RNN structure.

Reservoir Computing(RC) is a neural network architecture based on a Recurrent Neural Network(RNN). RNN is a class of artificial neural network which has been used for temporal pattern recognition. The feature of RNN is recurrent connections for obtaining dynamical reaction to input data. The process of RNN is same as below.

Process of RNN

1 **Generate dynamical reaction.** Input data perturbs the nodes for generating dynamical reactions with the weights on the recurrent connections.

2 **Preprocess dynamical reactions.** Compute the generated reactions by the activation function. In general, tanh and sigmoid functions are used for the activation function.

3 **Train the weights.** Compute the error between desired output and reactions. By the error, train the weights on the recurrent connections and in the readout.

The major algorithms for general RNN are BackPropagation Through Time(BPTT) [9] and Real-Time Recurrent Learning(RTRL) [10]. Most of the algorithms based on RNN use one of major algorithm or combined two algorithms. For the advanced algorithm of RNN, a number of global parameters are involved, so it can't be easily optimized. By a number of parameters, the costs of RNN is very high.

In the 2000s, RC is proposed as an advanced RNN to reduce the costs. Figure 2 represents the structure of Reservoir Computing. In the RC, the recurrent connection is called "reservoir".

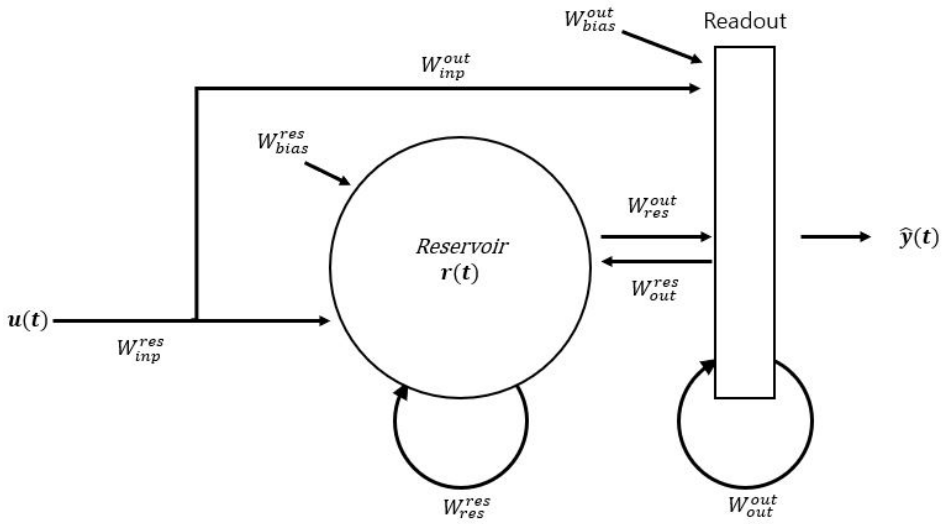


Figure 2: The general structure of Reservoir Computing. $\mathbf{W}^{res}$ are fixed random parameters, $\mathbf{W}_{out}$ are the parameter that we need to obtain by train.

Below equations are general Reservoir Computing model:

$$\mathbf{r}(t+1) = f(\mathbf{W}_{res}^{res}\mathbf{r}(t) + \mathbf{W}_{inp}^{res}\mathbf{u}(t) + \mathbf{W}_{out}^{res}\mathbf{y}(t) + \mathbf{W}_{bias}^{res}),$$
$$\hat{\mathbf{y}}(t+1) = \mathbf{W}_{res}^{out}\mathbf{r}(t+1) + \mathbf{W}_{inp}^{out}\mathbf{u}(t) + \mathbf{W}_{out}^{out}\mathbf{y}(t) + \mathbf{W}_{bias}^{out}.$$

$\mathbf{u}, \hat{\mathbf{y}}$ represent the input and output data vectors, $\mathbf{r}$ is the state of the reservoir. Each $\mathbf{W}_*^*$ are the parameters in each layers. $\mathbf{W}_*^{res}$ is the parameter to observe the behaviors of the reservoir, $\mathbf{W}_*^{out}$ is the parameter to interpret the behaviors in a readout layer.

The distinguishing feature of RC, compared to the general Artificial neural network(ANN), is the training. ANN generally trains all parameters, but RC modifies the parameter in readout layer. $\mathbf{W}_*^{res}$ keeps the initial set value even if the learning is progressed, $\mathbf{W}_*^{out}$ is trained in order to obtain the desired output.

The represented models of RC are LSM and ESN. LSM is proposed by Maass [3] to explore the computational capability of neural microcircuits in the brain [11]. The purpose of LSM is to develop biologically relevant learning models of spiking neural network with recurrent connections. The properties of LSM follows the constraints of biological neural networks. The probability that two neurons are connected depends on the distance between their positions.

The ESN was proposed by Jaeger [1], which uses an RNN-based reservoir consisting of discrete-time artificial neurons.

2.1 Echo State Network

We consider the input, reservoir, output as $\mathbf{u} \in \mathbb{R}^M, \mathbf{r} \in \mathbb{R}^N, \mathbf{y} \in \mathbb{R}^d$. The weights are collected in a matrix $\mathbf{W}_{in} \in \mathbb{R}^{N \times M}$ for the input weights, in an adjacency matrix $A \in \mathbb{R}^{N \times N}$ for the reservoir connections, and in a matrix $\mathbf{W}_{out} \in \mathbb{R}^{d \times (M,N,L)}$ for the readout.
The equation for ESN is described as follow:

$$\mathbf{r}(t) = \mathbf{f}\left(\mathbf{W}_{in}\mathbf{u}(t) + \mathbf{A}\mathbf{r}(t-1)\right), \tag{1}$$

$$\mathbf{y}(t) = \mathbf{f}^{out}\left(\mathbf{W}_{out}\left(\mathbf{u}(t), \mathbf{r}(t), \mathbf{y}(t-1)\right)\right), \tag{2}$$

where $\mathbf{f} = (f_1, ..., f_N)^T$ are reservoir's activation function(typically sigmoid function or tanh function),$\mathbf{f}^{out} = (f_1^{out}, ..., f_d^{out})^T$ are readout activation function, and $(\mathbf{u}(t), \mathbf{r}(t), \mathbf{y}(t-1)) = \left[\mathbf{u}(t)^T, \mathbf{r}(t)^T, \mathbf{y}(t-1)^T\right]^T$. $f_i(\mathbf{V}), f_i^{out}(\mathbf{v})$ are calculated in ith component of a vector $\mathbf{v}$.

The input matrix $\mathbf{W}_{in}$ is initialized by random distribution and we don't adjust the matrix as mentioned. Then, The difference between each network can be occurred by the random input matrix $\mathbf{W}_{in}$, but we don't need to deal with this problem because of the Echo State Property(ESP).

Definition 1 *Assume that input data is drawn in from a compact input space $U \subset \mathbb{R}^M$ and reservoir lies in a compact set $\boldsymbol{R} \subset \mathbb{R}^N$. Then the network has echo state property, if the network state $\boldsymbol{r}(t)$ is uniquely determined by any left-infinite input sequence. More precisely, this means that for every input sequence..., $\boldsymbol{u}(t-1), \boldsymbol{u}(t) \in U$, for all state sequences ..., $\boldsymbol{r}(t-1), \boldsymbol{r}(t)$ and ..., $\boldsymbol{r}'(t-1), \boldsymbol{r}'(t)$, it holds that $\boldsymbol{r}(t) = \boldsymbol{r}'(t)$.*

The condition for ESP is that a network state is determined by the left-infinite input sequence. That means, we need enough transient input data for converging to a stable network state.

To satisfy ESP , there is a sufficient condition(Proposition 1 [12]).

Proposition 1 *Assume a network with $f_i(v) = \tanh(v)$. Let the weight matrix $\boldsymbol{A}$ satisfy $\rho < 1$, where ρ is a spectral radius of $\boldsymbol{A}$. Then ESP holds for all input $\boldsymbol{u}$, for all reservoir $\boldsymbol{x}, \boldsymbol{x}' \in [-1, 1]^N$.*

So if the spectral radius of $\mathbf{A}$ is less than 1, we don't need to consider the random matrix $\mathbf{W}_{in}$.

To compute the weight $\mathbf{W}_{out}$, some methods are developed such as pseudo inverse and Singular Value Decomposition(SVD). In these methods, if the smallest singular value of reservoir $\mathbf{R}$ is close to zero, $\mathbf{R}$ does not have full column rank, resulting in ill-posed problem. To deal with this problem, some regularization methods, such as l_2 optimization(Ridge Regression-ESN [13]) and l_1 optimization(Lasso-ESN [14]), have been applied.

The algorithm of ESN is processed as below:

Algorithm of ESN

1 **Initialization.** Initialize the global parameter $\mathbf{A}, \mathbf{W}_{in}, \mathbf{r}(-\tau)$ where τ is the length of the transient input data. Rescale the spectral radius ρ of A to be less than unity.

2 **Make Reservoir.** Perturb the input data $\mathbf{u}(t)$ into reservoir state $\mathbf{r}(t)$ and construct the reservoir matrix $R = [\mathbf{r}_1, ..., \mathbf{r}_T]$ during training period T.

3 **Compute output.** Compute the network output $\hat{\mathbf{Y}} = [\hat{\mathbf{y}}(1), ..., \hat{\mathbf{y}}(T)]^T$ by (2).

4 **Derive output weight.** Observe the output weight $\mathbf{W}_{out}$ by optimization such as pseudoinverse and SVD. Then ESN is trained.

2.2 Modified versions of ESN

In this section, we introduce modified versions of ESN. Before the introduction, we define the parameters. $\mathbf{u}(t) \in \mathbb{R}^M$ is the input data and $\mathbf{y}(t) \in \mathbb{R}^d$ is the output data. M, d are the dimension of input and output space. $\mathbf{r}(t) \in \mathbb{R}^N$ is a reservoir state at time t and N is the number of nodes. $\mathbf{A} \in \mathbb{R}^{N \times N}$ is the sparse adjacency matrix which represents the connection between each other nodes. $\mathbf{W}_{in} \in \mathbb{R}^{N \times M}$ is a linear input weight matrix.

First one is ESN for the data reconstruction in [2]. Figure 3 shows what is the data reconstruction. The purpose of this example is recovering the lost system data $\tilde{\mathbf{y}}$ with the correlated system data $\tilde{\mathbf{u}}$.

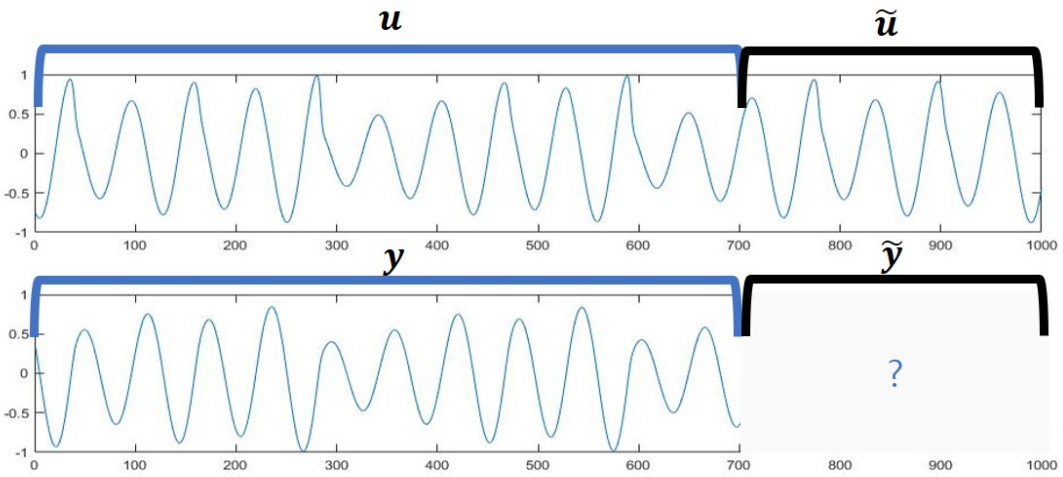


Figure 3: Data reconstruction. We find the Parameters(weights & biases) by training with input data $\mathbf{u}$ and output data $\mathbf{y}$, and We reconstruct $\tilde{y}$ by $\tilde{u}$.

The model of reservoir observer is described as below:

$$\mathbf{r}(t+1) = (1-\alpha)\mathbf{r}(t) + \alpha \tanh(\mathbf{A}\mathbf{r}(t) + \mathbf{W}_{in}\mathbf{u}(t) + \xi\mathbf{1}), \tag{3}$$

$$\hat{\mathbf{y}}(t) = \mathbf{W}_{out}\mathbf{r}(t) + \mathbf{C}. \tag{4}$$

The parameter $0 \leq \alpha \leq 1$ is a leakage rate that makes more accurate, and $\xi\mathbf{1} \in \mathbb{R}^N$ is the bias vector and ξ is the constant. $\mathbf{C} \in \mathbb{R}^d$ is a bias vector in readout layer. $\mathbf{A}$ is formed by sparse random Erdős-Rényi matrix in which the fraction of nonzero matrix element is $\frac{D}{N}$, so that the

average degree is D. We then rescale all elements of $\mathbf{A}$ so that the maximum magnitude of eigenvalues becomes scalar value ρ.

To compute the output weight W_{out}, they optimize the cost function$E(W_{out})$:

$$E(W_{out}) = \left\{ \sum_{k=1}^{K} \|\mathbf{W}_{out}\mathbf{r}(k\triangle t) + \mathbf{C} - \mathbf{y}(k\triangle t)\|^2 \right\} + \beta \left[\mathbf{Tr}\left(\mathbf{W}_{out}\mathbf{W}_{out}^T\right) \right], \tag{5}$$

where the $\|\mathbf{q}\|^2 = \mathbf{q}^T\mathbf{q}$ for a vector $\mathbf{q}$ and $\mathbf{Tr}(\mathbf{B}) = \sum_{i=1}^{n} b_{ii}$ is a trace of the square matrix $\mathbf{B} = [b_{ij}]_{n\times n}$. The second term of (5) is a regularization term included to avoid overfitting the output weight W_{out}, where β is the ridge regression parameter.

If the training is successful, the output $\hat{\mathbf{y}}$ yield a good approximation($\tilde{\mathbf{y}}$) to the desired output $\mathbf{y}$ after the training period($t > T$). Referring to equation 4:

$$\tilde{\mathbf{y}}(t) = \mathbf{W}_{out}^{*}\mathbf{r}(t) + \mathbf{C}^{*}, \tag{6}$$

where $\mathbf{W}_{out}^{*}$ and $\mathbf{C}^{*}$ denote the solutions for the minimizers of cost function $E(W_{out})$.

$$\mathbf{W}_{out}^{*} = \delta\mathbf{Y}\delta\mathbf{R}^T\left(\delta\mathbf{R}\delta\mathbf{R}^T + \beta\mathbf{I}\right)^{-1},$$

$$\mathbf{C}^{*} = -\left[\mathbf{W}_{out}^{*}\bar{\mathbf{r}} - \bar{\mathbf{y}}\right],$$

$$\bar{\mathbf{r}} = \frac{1}{K}\sum_{k=1}^{K}\mathbf{r}(k\triangle t),\ \bar{\mathbf{y}} = \frac{1}{K}\sum_{k=1}^{K}\mathbf{y}(k\triangle t),$$

where $\mathbf{I}$ is the identity matrix, $\delta\mathbf{R}$(Respectively, $\delta\mathbf{Y}$) is the matrix whose kth column is $\mathbf{r}(k\triangle t) - \bar{\mathbf{r}}$ (Respectively, $\mathbf{y}(k\triangle t) - \bar{\mathbf{y}}$).

The other version is a model for data prediction in [6].

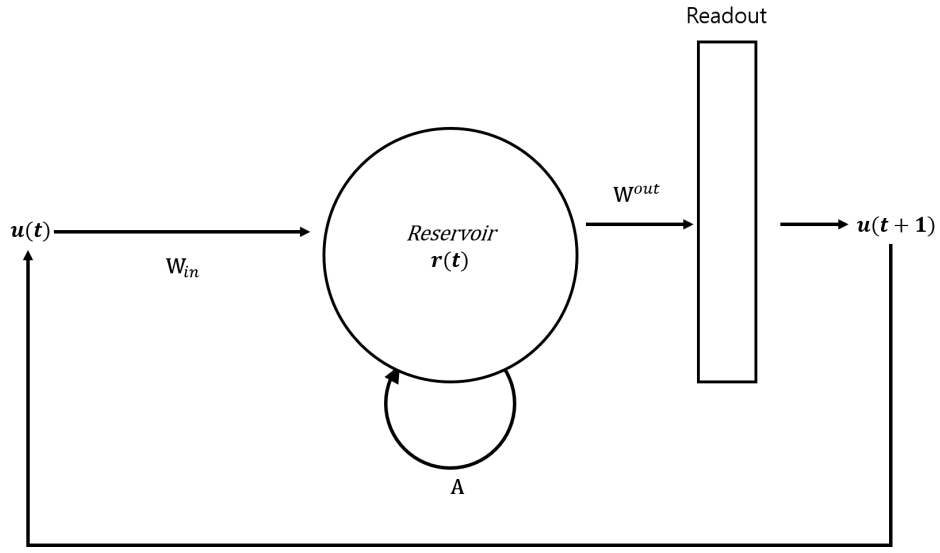


Figure 4: Structure of prediction after training period.

Figure 4 shows the structure of prediction. Data prediction is structurally different from reconstruction. In the case of reconstruction, input data can be accessed even after the training

period. Prediction use the output data obtained through the previous process as the input data after training period. In this example, the desired condition is that network output is a good approximation to $\mathbf{u}(t)$.

The network is given by following equations:

$$\mathbf{r}(t+1) = \tanh\left[\mathbf{A}\mathbf{r}(t) + \mathbf{W}_{in}\mathbf{u}(t)\right], \tag{7}$$

$$\mathbf{y}(t+1) = \mathbf{W}_{out}\left[\mathbf{r}(t)\right]. \tag{8}$$

In the Eq. (8),

$$\mathbf{W}_{out}\left[\mathbf{r}(t)\right] = \mathbf{P}_1\mathbf{r}(t) + \mathbf{P}_2\mathbf{r}^2(t), \tag{9}$$

where $\mathbf{P}_1$ and $\mathbf{P}_2$ are $d \times N$ matrices, and $\mathbf{r}^2(t)$ is the N dimensional vector whose jth component is $r_j^2(t)$.

For the prediction, parallelized reservoirs structure(Figure 5) is used. Let the spatiotemporal data be a one dimensional grid of size Q. Then the data is denoted by the vector $\mathbf{u}(t) = [u_1(t), ..., u_Q(t)]^T$. The Q variables $u_j(t)$ are split into g groups, each group consisting of q spatially contiguous variables such that $gq = Q$. These groups are denoted by $\mathbf{g}_i(t) = \left(u_{(i-1)q+1}(t), ..., u_{iq}(t)\right)^T$. Each group, $\mathbf{g}_i$, is predicted by a reservoir $\mathbf{R}_i$.

For constructing the reservoir, the contiguous region data, not just previous data, is used for the input data. Input data is described by $\mathbf{h}_i$, where $\mathbf{h}_i(t)$ contains the spatial points in ith group as well as from two contiguous regions of l spatial points on its left- and right-hand sides,$\mathbf{h}_i(t) = \left(u_{(i-1)q-l+1}(t), u_{(i-1)q-l+2}(t), ..., u_{iq+l}(t)\right)$. The j in u_j is taken modulo Q for preventing out of range.

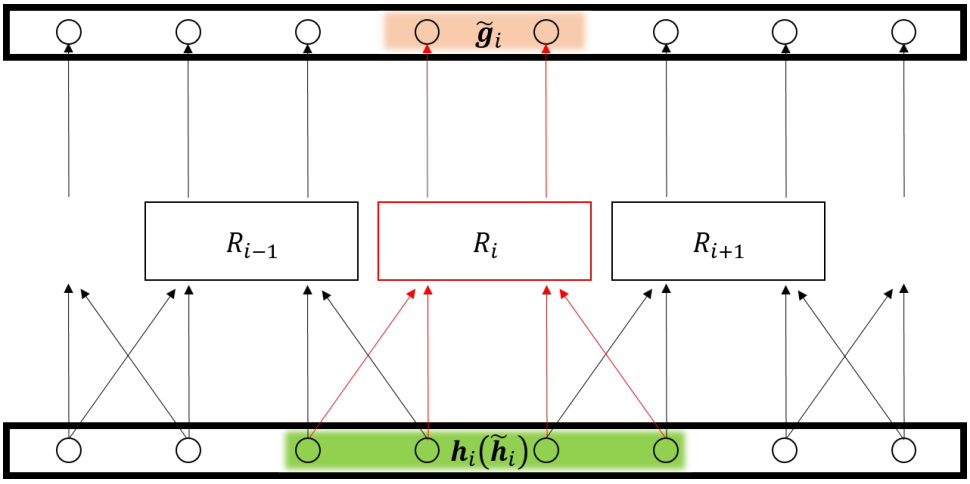


Figure 5: Structure of the parallelized reservoir.

III Dynamics based Reservoir Computing

In this section, we use the dynamical system, Kuramoto model, for constructing the reservoir. This network, Reservoir Computing based on Kuramoto model(RCK), is proposed by Choi in 2019 [5]. This method is RC that uses Kuramoto model [15], one of the simplest coupled oscillator, as a reservoir. Kuramoto model is a mathematical model describe synchronization. The most popular form of this model has the following equations:

$$\theta_i' = w_i + \frac{\lambda}{N}\sum_{j=1}^{N}\sin\left(\theta_j - \theta_i\right),\ i = 1, ..., N \tag{10}$$

where $\mathbf{w} = [w_1, w_2, ..., w_N]^T$ is natural frequency of each nodes, λ is coupling strength, and N is number of nodes.

To use Kuramoto model as dynamics, we need phase-locked state. After we perturb the input data to Kuramoto model, dynamics try to be the synchronized state. If the synchronized state is not a phase-locked state, then the state after perturbation can be quite different even if we perturb the same input data. That means, to obtain proper readout weight(W_{out}) is hard works.

For the phase locked-state, there are some conditions.

Theorem 1 *Let $D_0 \in (0, \pi)$, and $\theta = (\theta_1, ..., \theta_N)^T$ be the solution of equation 10 on a connected graph G with coupling strength and initial state that satisfy*

$$K > \frac{\sqrt{2}\sigma(\boldsymbol{w})}{L_* N \sin D_0} \text{ and } \varepsilon(\theta_0) < \frac{D_0^2}{2}.$$

Then we have

$$D\left(\theta(t)\right) < D_0 \text{ and } \lim_{t\to\infty}|\theta_i'(t)| = 0,\ i = 1, 2, ..., N.$$

Here $\sigma(\mathbf{w})$ is the total variance of natural frequencies ,$D\left(\theta(t)\right)$ is a phase diameter and $D_0 = D\left(\theta(0)\right)$:

$$\sigma(\mathbf{w}) := \left(\sum_{i=1}^{N}|\mathbf{w}_i|^2\right)^{\frac{1}{2}} \quad \text{and} \quad D\left(\theta(t)\right) := \max_{i,j}|\theta_i(t) - \theta_j(t)|$$

$\varepsilon(\theta_0)$ is the squared ℓ_2-norms of the phase and L_* is constant:

$$\varepsilon(\theta_0) := \sum_{i=1}^{N}|\theta_{i0}|^2 \quad \text{and} \quad L_* := \frac{1}{1 + \text{diam}(G)\,|E^c(G)|}$$

where $E^c(G) = V\times V - E(G)$ and $V, E(G)$ are vertex and edge set of graph G. $\text{diam}(G)$ represents the diameter of G, i.e., the longest length of the shortest path between pairs of vertices.

3.1 Reservoir Computing based on explosive Kuramoto model

There are various version of the Kuramoto model, we use the explosive Kuramoto model(equation 11).

$$\text{Explosive Kuramoto model}: \theta_i' = w_i + \frac{\lambda}{N}|w_i|\sum_{j=1}^{N}A_{ij}\sin\left(\theta_j - \theta_i\right), \tag{11}$$

where $\mathbf{A} = [A]_{N\times N}$ is adjacency matrix.

The different settings between popular form and explosive form are the coupling strength. In the case of coupling strength, popular form uses a constant(λ). However, the coupling strength of explosive form is proportional to each natural frequency($\lambda_i = \lambda\|w_i\|$).

The reason why we used the explosive form is the sensitivity of the coupling strength. To check the sensitivity of coupling strength, there is a measurement r, which is a Kuramoto order, for the synchronization described as follow:

$$re^{i\Phi} = \frac{1}{N}\sum_{j=1}^{N} e^{i\theta_j}, \tag{12}$$

where r represents the phase-coherence of the node of oscillator and Φ indicate the average phase.

In the case of r, we can check just coherence, not phase-locked state. So, [5] proposed another measurement variance order r_{var}.

$$r_{var} = \frac{1}{N}\sum_{j=1}^{N} \exp\left(-c\mathrm{var}_j\right),\ c > 0,$$

where var_j is the temporal variance of the $\theta_j'(t)$. c is a constant which represents the sensitivity to deviation from the phase-locked state and the range is $10^2 \le c \le 10^8$.

r_{var} is a measurement for desynchrony that sensitively shows a degree of deviation of oscillators from a steady frequency. If the Kuramoto model is in the phase-locked state, then the temporal variance var_j is kept close to 0.

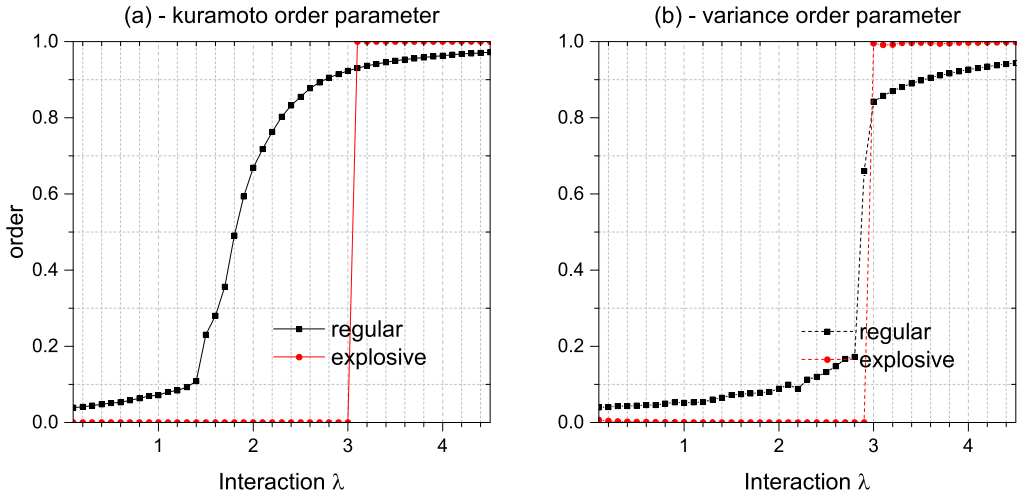


Figure 6: Phase-coherence order r and variance order r_{var} according to coupling strength.

Figure 6 shows the orders according to coupling strength. There are steep slopes for explosive Kuramoto model in two figures and the location of steep slopes are the same. That means, reservoirs start to synchronized as coherence and phase-locking at the same coupling strength.

If the start point of synchronization is different, then the strengths of each synchronization are different. For the fine reservoirs, the strength of synchronization should be properly strong.

RC based on explosive Kuramoto model(RCK) differs from the ESN in the process of making a reservoir. While ESN makes a reservoir by the Eq. (1), RCK directly perturbs the target nodes and then obtains the reaction of a reservoir in the process of synchronization. The details are as follows.

Process of Reservoir Computing based on explosive Kuramoto model

1 **Initialize the variables.** Initialize random variables, $\theta(0)$ and w.

2 **Choose the coupling strength.** Calculate the coherence order r for each coupling strength K in $[1.0, 4.0]$ and choose the proper coupling strength.

3 **Observe the reaction.** Add the value of input data $\mathbf{u}(t) = [u_1(t), u_2(t), ..., u_M(t)] \in \mathbb{R}^M$ to each chosen M nodes as the perturbation. The reactions are observed at regular interval St during certain time T_{res}.(Figure. 7)

4 **make reservoir matrix.** The reactions are stored as the reservoir matrix. And then, we repeat the 3rd process with $\mathbf{u}(t+1)$.

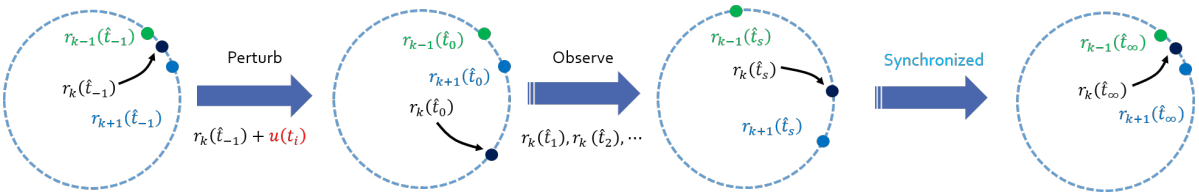


Figure 7: Process of step 3. Perturb some nodes by the input data. And then, Nodes try to be synchronized. We observe the movement of synchronization until enough time for synchronization.

In the 3rd process, details are described in Figure 7. First, we choose the M nodes for perturbation. And then, we perturb the input data $u(t_i)$ to reservoir state directly:

$$\hat{r}_k(t) \leftarrow \hat{r}_k(t) + u_k(t) \text{ where } k = 1, 2, ..., M,$$

and $\hat{r}_k(t), u_k(t)$ are the components of reservoir states and input data. By the perturbation, the synchronization is broken, so the reservoir try to be synchronized. During the reservoir reacts to be synchronized, we observe the movement of nodes and save the movement with this form:

$$\mathbf{r}(t_i) = \left[\tilde{r}_1(\hat{t}_1), \tilde{r}_2(\hat{t}_1), ..., \tilde{r}_N(\hat{t}_1), \tilde{r}_1(\hat{t}_2), ..., \tilde{r}_N(\hat{t}_2), ..., \tilde{r}_N(\hat{t}_K)\right]^T$$

where $\tilde{r}_k(\hat{t}_j) = r_k(\hat{t}_j) - r_k(\hat{t}_{j-1})$, $St = \hat{t}_j - \hat{t}_{j-1}, T_{res} = \hat{t}_K$. The dimension of reservoir state at time t_i is $N \times T_{res}/St$.

Also, we need to adjust the scale of the input data. In the case of the kuramoto model, a trigonometric function is used, so if the input data is $2n\pi + \theta$, the result is the same when

the input data is θ. Also, since the synchronization time varies depending on the frequency of each node, we can't randomly select nodes to be perturbed. In this work, we rescale the data into $[-1, 1]$ for maintaining the synchronization of RCK, and perturb the specific nodes whose frequency is close to 1.

3.2 Application

We compare two types of reservoir computing, ESN and RCK. The equation of ESN is almost same as a reservoir observer, but we don't use the bias term $\mathbf{C}$. Also, when we minimize the cost function(Eq. (5)), we adjust $\beta = 0$. So the cost function is equal to

$$\left\{ \sum_{k=1}^{K} \|\mathbf{W}_{out}\mathbf{r}(k\triangle t) - \mathbf{y}(k\triangle t)\|^2 \right\},$$

and the minimizer $\mathbf{W}_{out}$ is obtained in a closed-form [16] as,

$$\mathbf{W}_{out} = (\mathbf{Y}\mathbf{R}^T)(\mathbf{R}\mathbf{R}^T)^{-1} \text{ where } Y = [\mathbf{y}(\triangle t), \mathbf{y}(2\triangle t), ...], R = [\mathbf{r}(\triangle t), \mathbf{r}(2\triangle t), ...],$$

with $\mathbf{A}^{-1}$ is the Moore-Penrose pseudo inverse of $\mathbf{A}$.

For the comparison, we use such examples, reconstruction, prediction.

A. Data Reconstruction

First examples of reconstruction are typical chaotic systems, Rössler system and Lorenz system. Left equation represents the Rössler system, and right one is Lorenz system.

$$\begin{aligned} \frac{dx}{dt} &= -y - z, & \frac{dx}{dt} &= \sigma(y - x), \\ \frac{dy}{dt} &= x + ay, & \frac{dy}{dt} &= \rho x - y - xz, \\ \frac{dz}{dt} &= b + z(x - c), & \frac{dz}{dt} &= -\beta z + xy, \end{aligned}$$

where $a = 0.5, b = 2.0, c = 4.0, \sigma = 10, \beta = 8/3, \rho = 28$.

For this experiment, we utilize the parameters as follow. ESN parameters are from [2].

For the ESN,

number of reservoir nodes:	$N = 400$,
spectral radius:	$\rho = 1.0$,
average degree:	$D = 20$,
scale of input weights:	$\sigma = 1.0$,
bias constant:	$\xi = 1.0$,
leakage rate:	$\alpha = 1.0$,
time interval:	$\triangle t = 0.1$,
length of training phase:	$T = 260$.

For the RCK,

number of reservoir nodes:	$N = 60,$
coupling strength:	$\lambda = 2.3,$
time interval for saving reservoir:	$St = 0.1,$
Observation time for each input:	$T_{Res} = 1,$
time interval of data:	$\triangle t = 0.1,$
length of training phase:	$T = 260.$

We use the Root Mean Square Error(RMSE Eq. (13)) to check the performance.

$$\text{RMSE} = \sqrt{\sum_{i=1}^{n} \frac{(\tilde{y}_i - \hat{y}_i)^2}{n}} \text{ where } \hat{y}. \text{ is the reconstructed data} \tag{13}$$

Figure 8, 9 show the results of reconstruction. We test 1,000 time units for the Rössler system, and 500 time units for the Lorenz system. There is no difference in Figures. In the case of RMSE, ESN has better results than RCK.

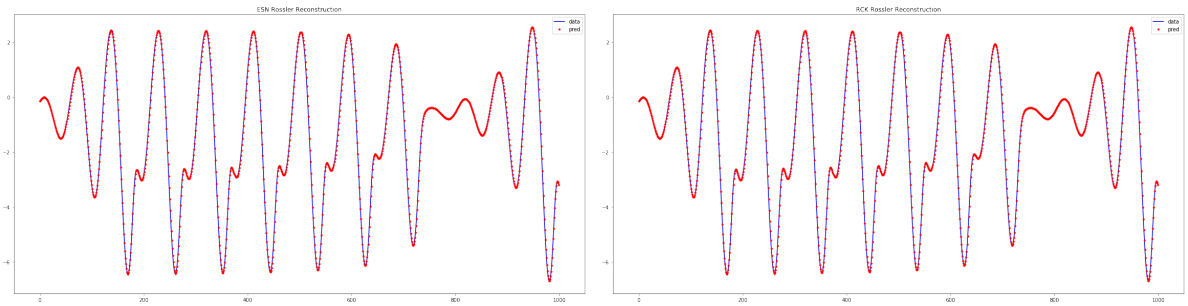


Figure 8: Reconstruction the Rössler y data by x data. Left is reconstructed by ESN, right is by RCK. RMSE is 3.698×10^{-4} by ESN and 8.134×10^{-4} by RCK.

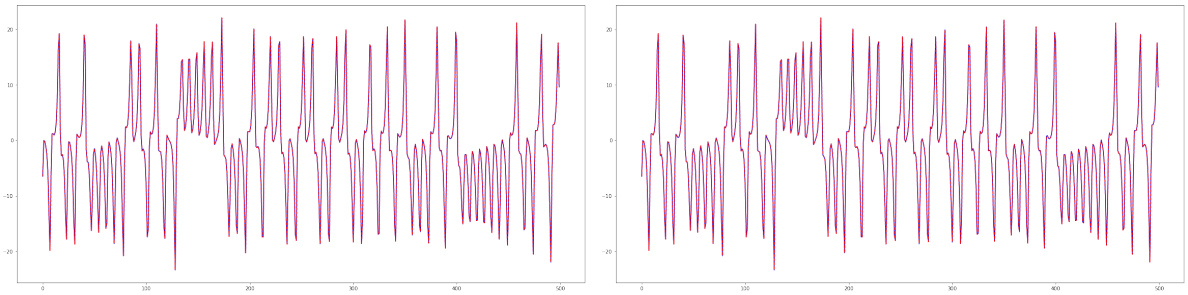


Figure 9: Reconstruction of the Lorenz y data by x data. Left is reconstructed by ESN, right is by RCK. RMSE is 3.368×10^{-3} by ESN and 1.950×10^{-2} by RCK.

Next example is the Kuramoto-sivashinsky model.

$$y_t = -yy_x - y_{xx} - y_{xxxx}. \tag{14}$$

For this experiment, we impose spatially periodic boundary conditions. We set $y(x+L,t) = y(x,t)$ on a domain $x \in (0, L)$ of size $L = 22$ and integrate the Eq. (14) from randomly chosen initial condition. For the integration, we use the spectral method with spaced grid of size $Q = 65$. This system data is denoted by $\mathbf{y}(t) = (y_1(t), y_2(t), ..., y_Q(t))$ with $y_i(t) = y(i\triangle x), \triangle x = L(Q-1)$.

For the experiment, we utilize the parameters as follow.
For the ESN,

$$N_{ESN} = 3000, \quad \rho = 0.9, \quad D = 60, \quad \sigma = 0.5$$
$$\xi = 0.0, \quad \alpha = 0.3, \quad \triangle t = 0.25, \quad T = 15000.$$

For the RCK,

$$N_{RCK} = 300, \quad \lambda = 2.2, \quad St = 0.1$$
$$T_{res} = 1 \quad \triangle t = 0.25, \quad T = 15000.$$

For the KS equation reconstruction, input data is a vector containing M out of the Q time series in $\mathbf{y}(t)$.

We set the input data $\mathbf{u}(t) = (u_1(t) = y_2(t), u_2(t) = y_{32}(t))$ with $M = 2$ and other $y_i(t)$'s are the output data. The goal of this experiment is to reconstruct the data $\mathbf{d}(t) = \mathbf{y}(t) \setminus \mathbf{u}(t)$. In the Figure 10, we can see the result of ESN and RCK. Left figures are for the ESN, and right ones are for RCK. Above figures shows the original figure of the KS equation data, and center row shows the reconstructed data by ESN and RCK. Right figure is more clear and similar than the left one. Below figure in the Figure 10 shows the difference between original and reconstructed data. The blurred figure means a better reconstruction, and RCK figure seems more blurry.

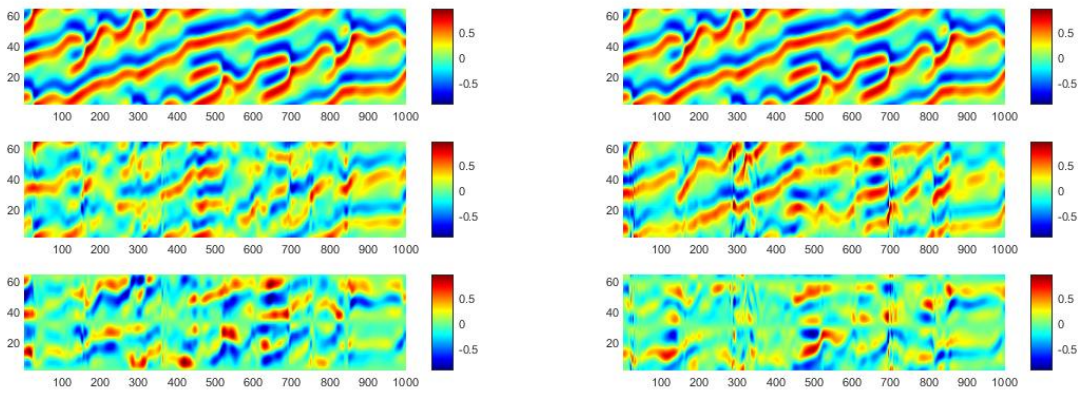


Figure 10: Reconstruction of the Kuramoto-sivashinsky equation data. L: reconstructed by ESN, R : reconstructed by RCK.

Correlation and RMSE were used as measurements to numerically indicate this difference. The calculations are as follows.

$$\text{Correlation}(C) = \frac{\sum_{i,t}\left(d_i(t) - \langle d\rangle\right)\left(\hat{d}_i(t) - \langle \hat{d}\rangle\right)}{\sqrt{\left(\sum_{i,t}(d_i(t) - \langle d\rangle)^2\right)\left(\sum_{i,t}(\hat{d}_i(t) - \langle \hat{d}\rangle)^2\right)}},$$

$$\text{RMSE} = \sqrt{\frac{\sum_{i,t}\left[d_i(t) - \hat{d}_i(t)\right]^2}{\sum_{i,t}\left[d_i(t)\right]^2}},$$

where $\hat{d}_i(t)$ is the reconstructed data.

It can be seen as a good result when Correlation is high and RMSE is low, and RCK is showing better results.

	ESN	RCK
Correlation	0.7574	0.7943
RMSE	0.6535	0.6052

Table 1: Correlation and RMSE when $M = 2$.

In the case of RCK, the larger the M, the worse the result (Figure 11). It seems to come from the structural problem of RCK. When the dimension of input data increases, the number of nodes to be perturbed increases. This causes the synchronization state to break, and prevents the Kuramoto model from fully functioning.

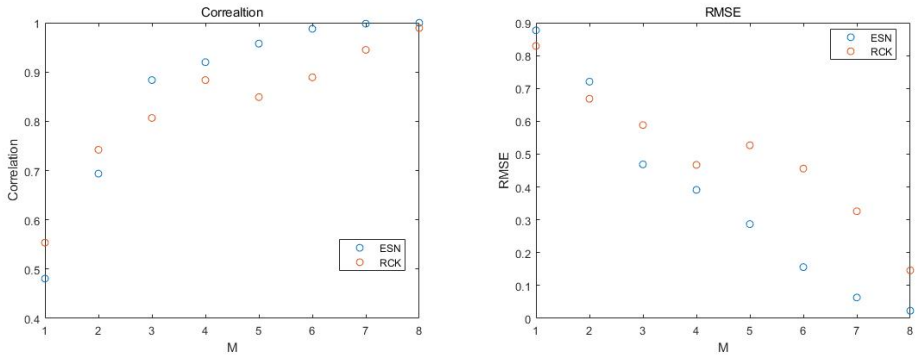


Figure 11: The correlation(L) and RMSE(R) for each dimension(M) of input data.

B. Data Prediction

Next example is data prediction. The structure is same as [6], but since the data is 1-dimensional, it is not necessary to use a parallelized reservoir.

The parameters are mostly the same as those used for reconstruction, and the difference is that the number of ESN nodes is 600, and the length of training phase is increased from 260s to 400s.

a. Prediction with original system data

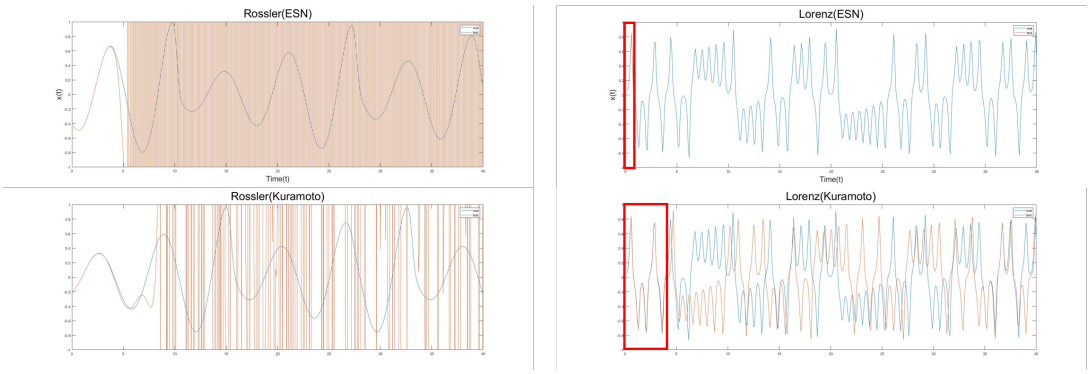


Figure 12: Predict the y data with original x data of system

Figure 12 is the result of prediction for x data of Lorenz system and Rössler system. RCK makes a better prediction, but even that is not long. As the prediction progresses, we desired the network output is a good approximation of x data, but it doesn't. The problem is that the error is stacked when the process is repeated. This stacked error can't be covered by $\mathbf{W}_{out}$ because $\mathbf{W}_{out}$ is trained by clear input data.

b. Prediction with noised system data

Instead of using the original data $u(t)$, noised data $\hat{u}(t)$ was used. We use Gaussian distribution for noise, and the deviation (δ) was set to 0.01(Eq. (15)).

$$\hat{u}(t) = u(t) + \delta \times w \times u(t) \text{ where } w \text{ is Gaussian noise.} \tag{15}$$

We expect that the length of prediction will be increased by the noise. Below figure 13 is a result of prediction with noise.

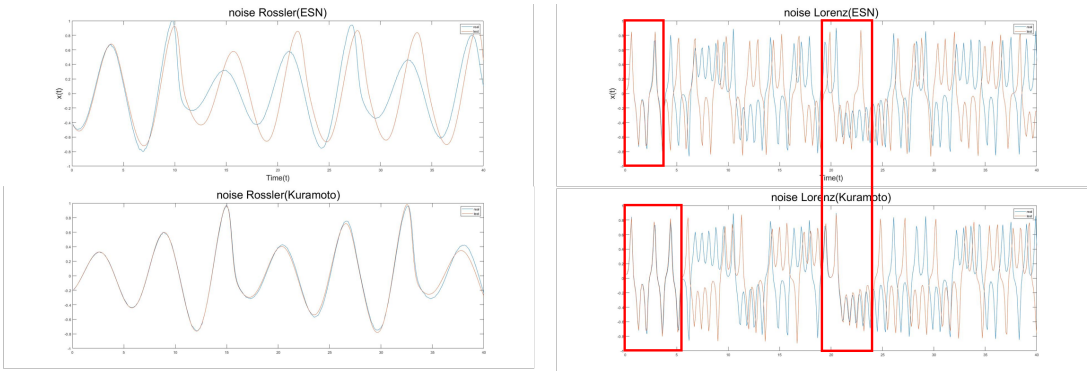


Figure 13: Predict the y data with noised x data of system

As a result, the result was slightly improved in both RC model. In the case of Rössler system, the out of range part is disappeared. In particular, we could get better results close to the original data for the Rössler system. In the case of Lorenz system, the length of prediction is increased a little. There are additional predictable parts in RCK figure, but not for the ESN.

3.3 Frequency sensitivity

When we use the Rössler and Lorenz system for Kuramoto model, we perturb the first node $r_1(t)$. Since we use the same parameters for those systems and there is no problem with the task, we don't need to consider which node is perturbed. However, if the natural frequency and the number of nodes are changed, a problem can occur depending on which node is perturbed. The below table shows the relative error when we perturb the nodes which have each frequency.

Frequency	-2.1384	-1.9330	-1.7947	-0.8396	-0.0825
RelErr	6.2758E04	0.0067	9.7604E-05	5.3863E-06	1.8278E03
Frequency	0.1001	0.1240	0.1873	0.8351	1.7119
RelErr	735.6910	2.1355E07	8.9962E-07	1.1517E-05	1.3873E-04

Table 2: Relative error when we perturb the node which has each natural frequency.

The relative error is calculated as:

$$\text{RelErr} = \frac{\sum (\hat{y} - y)^2}{\sum y^2} \tag{16}$$

where $\hat{y}$ is a network output and y is a desired output. When we choose the node whose frequency is 0.1873, relative error is smallest. In the case of relative error for reconstruction, if the natural frequency is very large or small, it seems that there is a problem. The reason that we think is a synchronization state.

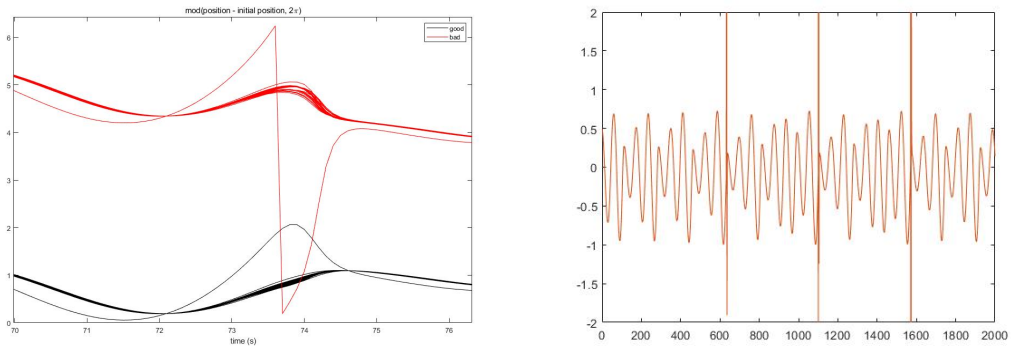


Figure 14: L:The position difference between an initial position and after perturbing, R: wrong reconstruction.

Left figure on Figure. 14 shows the position difference between an initial position and after perturbing. The red line represents the poor reconstruction case, and the black line is the well reconstruction case. If we see the red line, we can think that the reason for the problem is that a particular node rotates once. In that section, this model does not play a proper role for the reservoir, that means, we can't reconstruct well when the nodes rotate for synchronization.

The reason for the rotation seems to be that the interval we perturb is insufficient for synchronization. For this paper, we set the Observation time T_{Res} to 1, but in this case, it seems

to be out of synchronization in 1. So, we increase the time T_{Res} to 10, and also we increase the save time interval St from 0.1 to 1 to make the same reservoir size.

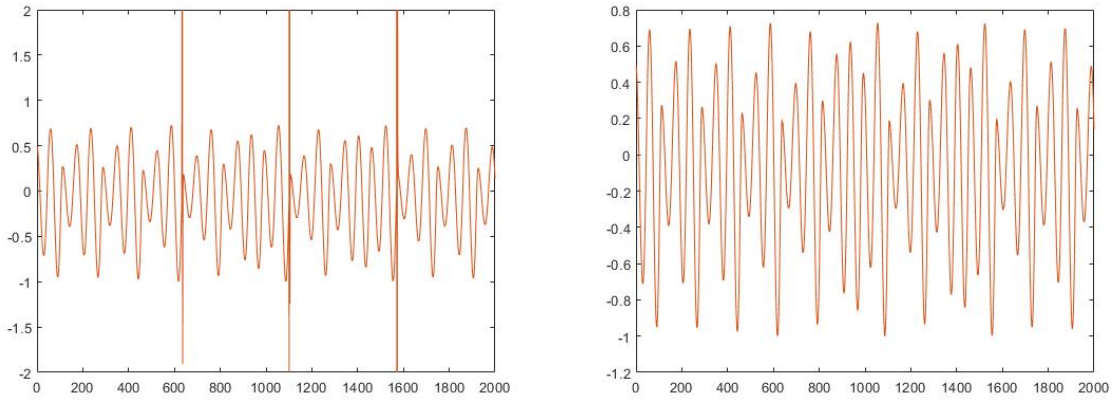


Figure 15: Rössler system reconstruction when L : Observation time $T_{Res} = 1$, R : $T_{Res} = 10$.

Figure 15 shows the result of the time increment. When the time is increased to 10, the reconstruction become possible in all sections. By these experiments, for good results, natural frequency and perturbation intervals need to be properly considered.

IV Dynamics based Reservoir Computing with Concatenation

We also test the RCK by capacity. Capacity is a measurement to evaluate the performance of reservoir, which is published in 2012 [17]. The purpose of the capacity is to quantify the different models in which information can be processed by such systems. Capacity is determined by the error of each order, and the calculation process is as follows:

$$C_T[X,z] = 1 - \frac{\min \mathrm{MSE}_T[\hat{z}]}{\langle z^2 \rangle_T},$$

where $\hat{z}, z$ are the network output function, desired output function as Eq. (2) and $X(t)$ is the reservoir state $X(t) = [x_1(t), ..., x_N(t)]^T$. MSE is a Mean Square Error which is calculated as:

$$\mathrm{MSE}_T[\hat{z}] = \frac{1}{T}\sum_{t=1}^{T} \left(\hat{z}(t) - z(t)\right)^2,$$

and T is the length of data. $\langle z^2 \rangle_T$ is described as:

$$\langle z^2 \rangle_T = \frac{1}{T}\sum_{t=1}^{T} z(t)^2.$$

In this case, there is property for the capacity.

Proposition 2 *For any function $z(t)$, and any set of recorded data $X(t)$, $t = 1, ..., T$ the capacity can be expressed as*

$$C_T[X,z] = \frac{\sum_{ij} \langle z x_i \rangle_T \langle x_j x_i \rangle_T^{-1} \langle x_j z \rangle_T}{\langle z^2 \rangle_T}.$$

and

Proposition 3 *For any z, and any set of x, the capacity is normalized according to:*

$$0 \leq C_T[X,z] \leq 1,$$

where $C_T[X,z] = 0$ if the dynamical system is unable to reconstruct z, and $C_T = 1$ if the system can reconstruct perfectly.

This is a partial capacity of specific function z. To qualify the performance of the reservoir, the input data is set to uniform distribution over the interval $[-1, 1]$, and the output is set to finite products of normalized Legendre polynomials. We need to calculate high order polynomials and finite products of polynomials. The total capacity of system is:

$$C_{\mathrm{TOT}} = \sum_{d,i} C_T[X, P_d(t-i)],$$

where $P_d(t)$ is a desired polynomial of uniform distribution,and d, i are order of Legendre polynomial and delay.

In order to calculate the capacity, all of the high-order orders must be calculated. However the results in the paper are excluded the high order Legendre polynomial($d > 5$) because the capacity for high order case is very small.

4.1 limitation of RCK

The result of capacity is shown by Figure 16. In this figure we compare two models according to reservoir size. The reservoir size of ESN is the same with the number of nodes and the size of RCK is the same as 10 times of the number of Nodes. The capacity of ESN is increased according to a size of reservoir, but the capacity of RCK is not. For the RCK, the capacity is increased until the size of reservoir is 400, but there is no increment after 400 reservoir size. Even if we increase reservoir size more than 400 reservoir size, we can't expect the performance improvement.

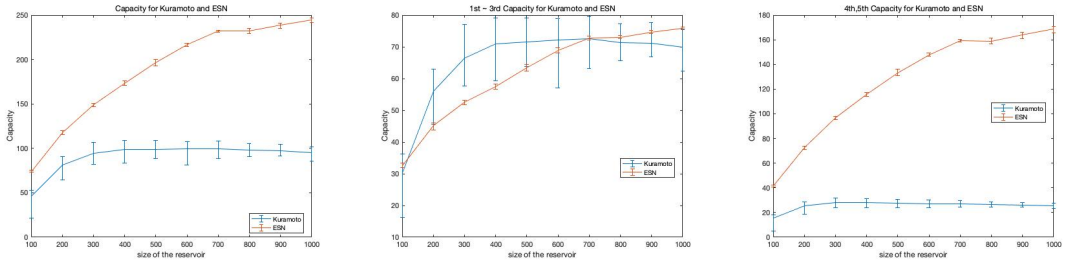
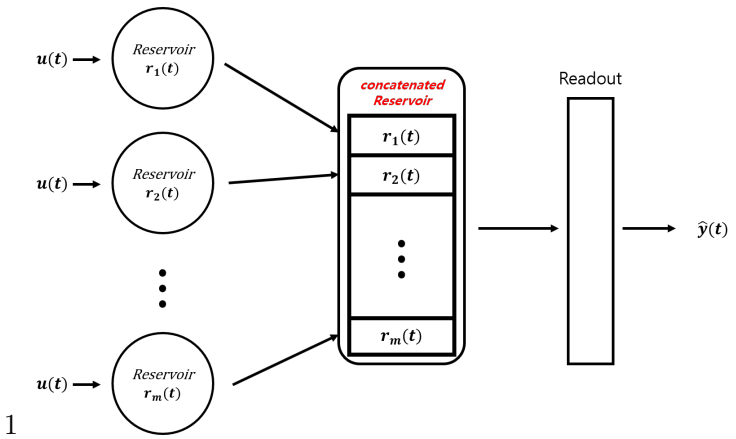


Figure 16: L : Total capacity, C : 1∼3 order capacity, and R: 4∼5 order capacity for ESN, RCK according to size of reservoirs.

When we see the center and right figure in Figure 16, the problem with RCK is revealed. Total capacity for $1 \sim 3$ order polynomial of RCK is similar to the ESN. However, the case of $4 \sim 5$ order, the difference in capacity is very large. This is very insufficient compared to ESN. The difference in capacity can be seen as appearing in high order, which means that RCK is weak for high order.

Also, there is a problem with cost time. When we observe the reservoir for reconstruction, the cost for ESN with 400 nodes is about 6s, but RCK takes 60∼70 seconds. The ESN reservoir is constructed by the nonlinear equation, but RCK is constructed by iterative method for system. Since these problems, we propose an enhanced model which is constructed by concatenation.

4.2 Reservoir computing with concatenation



1

Figure 17: Structure of Concatenated Reservoir Computing.

Figure 17 shows the structure of concatenated RC. The difference between original method and concatenation is making reservoir. First, we initialize m reservoir states with N nodes by different global parameters(W_{in}, A, w). And then, we perturb the input data to m reservoirs, we have m different reservoirs $\mathbf{r}_1(t), \mathbf{r}_2(t), ...\mathbf{r}_m(t) \in \mathbb{R}^{N\times T}$ where T is the size of input data. We concatenate those m reservoirs as a one reservoir $\mathbf{R} = [\mathbf{r}_1(t), \mathbf{r}_2(t), ...\mathbf{r}_m(t)] \in \mathbb{R}^{(N\times m)\times T}$.

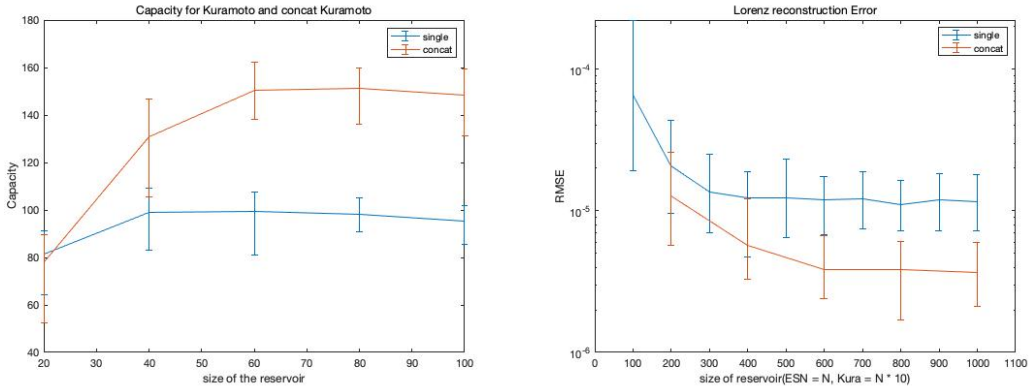


Figure 18: Capacity(L) and Lorenz reconstruction error(R) for RCK, concatenated RCK according to size of reservoirs.

The result of concatenation is shown in Figure 18. Single is just RCK and concat means concatenated RCK by 2 RCKs with same nodes. The limitation of capacity according to number of nodes is increased form 40 to 60. Also, the capacities are increased. By the concatenation, we can improve performance and overcome limitations.

4.3 Application

We apply the concatenation to RCK and ESN.

A. Mackey-Glass equation

The task to check performance improvement is the task1 in [5]. For this task, Mackey-Glass equation(MG) [18] is used. MG data which is scaled to $[-1, 1]$ is used as input data, and the output data is summation of delayed data(Eq. (17)).

$$y(t) = \frac{1}{m}\sum_{k=1}^{m}(ax(t-k) + bx(t-k)^2 + cx(t-k)^3), \tag{17}$$

where x is the MG data and a, b, c are some nonzero constants. In this case, we set $a = 1, b = 2, c = 5$.

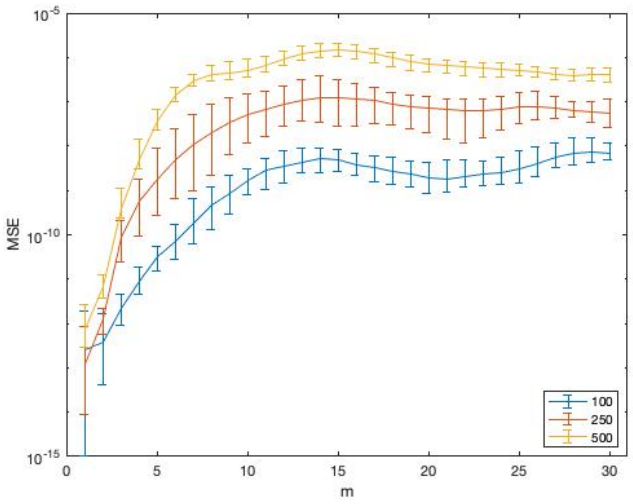


Figure 19: MSE of task1 in [5] according to m.

The result of task1 is shown in Figure 19. In the Figure 19 the yellow line is made by one reservoir with 500 nodes, and blue line is made by 5 reservoirs with 100 nodes, red one is by 2 reservoirs with 250 nodes. We can get the better results when we concatenate more reservoirs.

B. Reconstruction

We compare the Lorenz system reconstruction error for single RC and concatenated RC. We tested random 100 cases for each size of reservoirs and we compare the same reservoir size of RCK and ESN. The concatenated RC is constructed by 2 different reservoirs with same nodes.

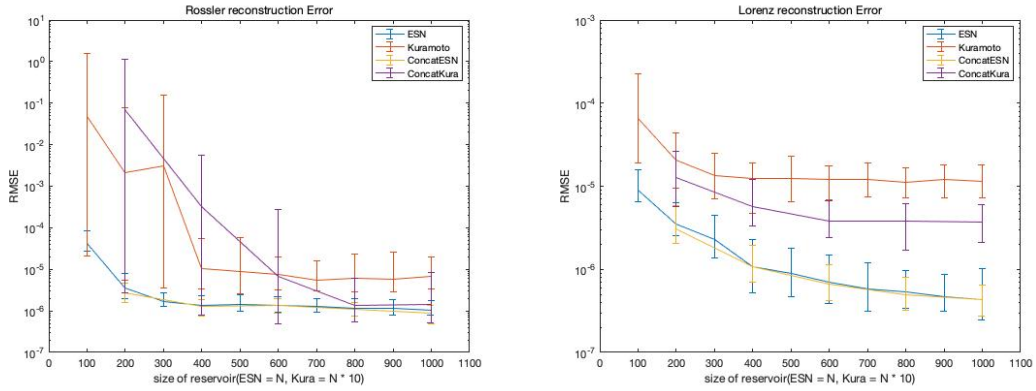


Figure 20: (L): Rössler RMSE, (R):Lorenz RMSE according to the reservoir size.

The figure 20 shows the result of reconstruction error according to a size of reservoirs. In the case of RCK, there is an improvement for the reconstruction. RMSE for Rössler reconstruction has a stability problem, but the minimum RMSE is decreased.

In the case of ESN, there is no difference by the concatenation. We think that the reason is ESP. Assume that there is two different ESNs $\mathbf{r}_1$ with $\left(\mathbf{A}_1, \mathbf{W}_{in}^1\right)$ and $\mathbf{r}_2$ with $\left(\mathbf{A}_2, \mathbf{W}_{in}^2\right)$. Let the concatenated ESN is $\mathbf{R} = [\mathbf{r}_1, \mathbf{r}_2]$. We set the adjacency matrix $\tilde{\mathbf{A}}$ and input weight $\tilde{\mathbf{W}}_{in}$ of single ESN $\tilde{\mathbf{R}}$ as below.

$$\tilde{\mathbf{A}} = \begin{bmatrix} \mathbf{A}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_2 \end{bmatrix} \quad \text{and} \quad \tilde{\mathbf{W}}_{in} = \begin{bmatrix} \mathbf{W}_{in}^1 & \mathbf{0} \\ \mathbf{0} & \mathbf{W}_{in}^2 \end{bmatrix}.$$

Since $\tilde{\mathbf{A}}$ and $\tilde{\mathbf{W}}_{in}$ are block diagonal matrices, so each blocks are independent. Then, the calculations of $\mathbf{R}$ and $\tilde{\mathbf{R}}$ are same. And, the spectral radius $\tilde{\rho}$ of $\tilde{\mathbf{A}}$ is $\max\left(\mathbf{A}_1, \mathbf{A}_2\right)$. If we set the spectral radius of $\mathbf{A}_1, \mathbf{A}_2$ are less than 1, then $\tilde{\rho}$ is also less than 1. So, concatenated ESN $\mathbf{R}$ has echo state property and this results that Single ESN and concatenated ESN converge to the same state.

V Conclusion

We introduce the two types of reservoir computing and compare them. For most tasks, Echo state network(ESN) has better performance, but Reservoir computing based on Kuramoto model(RCK) has a strength in the case of prediction. ESN has a stable result by the Echo state property, but RCK has no such property for stability. Also, RCK has a limitation according to the number of nodes, but ESN doesn't. In the case of RCK, there have been many problems with parameter setting because research on parameters has not been conducted much. Performance comparison according to parameters for synchronization of Kuramoto model also seems necessary.

In section 4, we propose enhanced reservoir computing with concatenation for the limitation. We split the nodes and perturb each small reservoirs. RCK has improved performance by the concatenation, but ESN doesn't. We think that the reason why ESN has no effect is the echo state property. Echo state property provide the stability regardless of number of nodes, random parameters which are satisfied the condition of property. For the RCK, performance can be improved by merging several, and the time required for synchronization can be reduced through parallelization.

In this paper, we use the pseudo inverse to obtain output weight W_{out}. As mentioned in section 2, pseudo inverse has ill-posed problem. To avoiding this ill-posed problem, we need to apply the other methods such as lasso-ESN, and RR-ESN. These methods can be applied RCK not just ESN.

References

[1] H. Jaeger and H. Haas, "Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication," *science*, vol. 304, no. 5667, pp. 78–80, 2004.

[2] Z. Lu, J. Pathak, B. Hunt, M. Girvan, R. Brockett, and E. Ott, "Reservoir observers: Model-free inference of unmeasured variables in chaotic systems," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 27, no. 4, p. 041102, 2017.

[3] W. Maass, T. Natschläger, and H. Markram, "Real-time computing without stable states: A new framework for neural computation based on perturbations," *Neural computation*, vol. 14, no. 11, pp. 2531–2560, 2002.

[4] N. Crook, "Nonlinear transient computation," *Neurocomputing*, vol. 70, no. 7-9, pp. 1167–1176, 2007.

[5] J. Choi and P. Kim, "Critical neuromorphic computing based on explosive synchronization," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 29, no. 4, p. 043110, 2019.

[6] J. Pathak, B. Hunt, M. Girvan, Z. Lu, and E. Ott, "Model-free prediction of large spatiotemporally chaotic systems from data: A reservoir computing approach," *Physical review letters*, vol. 120, no. 2, p. 024102, 2018.

[7] Y. Kuramoto, "Diffusion-induced chaos in reaction systems," *Progress of Theoretical Physics Supplement*, vol. 64, pp. 346–367, 1978.

[8] G. Sivashinsky, "Nonlinear analysis of hydrodynamic instability in laminar flames—i. derivation of basic equations," *Acta astronautica*, vol. 4, pp. 1177–1206, 1977.

[9] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," California Univ San Diego La Jolla Inst for Cognitive Science, Tech. Rep., 1985.

[10] R. J. Williams and D. Zipser, "A learning algorithm for continually running fully recurrent neural networks," *Neural computation*, vol. 1, no. 2, pp. 270–280, 1989.

[11] W. Maass and H. Markram, "On the computational power of circuits of spiking neurons," *Journal of computer and system sciences*, vol. 69, no. 4, pp. 593–616, 2004.

[12] H. Jaeger, "The "echo state" approach to analysing and training recurrent neural networks-with an erratum note," *Bonn, Germany: German National Research Center for Information Technology GMD Technical Report*, vol. 148, no. 34, p. 13, 2001.

[13] X. Dutoit, B. Schrauwen, J. Van Campenhout, D. Stroobandt, H. Van Brussel, and M. Nuttin, "Pruning and regularization in reservoir computing," *Neurocomputing*, vol. 72, no. 7-9, pp. 1534–1546, 2009.

[14] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 58, no. 1, pp. 267–288, 1996.

[15] Y. Kuramoto, *Chemical oscillations, waves, and turbulence*. Courier Corporation, 2003.

[16] D. W. Marquardt and R. D. Snee, "Ridge regression in practice," *The American Statistician*, vol. 29, no. 1, pp. 3–20, 1975.

[17] J. Dambre, D. Verstraeten, B. Schrauwen, and S. Massar, "Information processing capacity of dynamical systems," *Scientific reports*, vol. 2, no. 1, pp. 1–7, 2012.

[18] L. Glass and M. C. Mackey, "Pathological conditions resulting from instabilities in physiological control systems," *Annals of the New York Academy of Sciences*, vol. 316, no. 1, pp. 214–235, 1979.